Algorithms and Data Structures 2
CS 1501

Spring 2022

Sherif Khattab

ksm73@pitt.edu

(Slides are adapted from Dr. Ramirez's and Dr. Farnan's CS1501 slides.)

Announcements

- Upcoming deadlines:
  - Homework 11 due on 4/11
  - Lab 11 due on 4/15
  - Homework 12 due on 4/18
  - Assignment 3 and 4 due on 4/18

Previous lecture …

- Integer Multiplication
  - Karatsuba's algorithm
    - Divide and conquer
    - Reduce the number of subproblems

CourseMIRROR Reflections (most confusing)

- The concept of a minimum st-cut was most confusing today.
- how to find st-cut
- Which edges are counted for min ST cuts? What does it have to do with being reachable by S
- when talking about max flow, is it the flow over each edge or do you add up the flows of each edge on the path?
- Is there any connection between articulation points (if there are a direct path between two articulation points) and st-cut path. It seems as similar, as the whole graph can to maintain being connected/hold max flow
- It’d be nice to see a formalized algorithm for priority first search
- karatsuba is still confusing as to what exactly is happening.
- The multiplication algorithms and their runtime
- euclids algorithm

CourseMIRROR Reflections (most interesting)

- Seeing the difference between gradeschool multiplication and karatsuba's multiplication
- Optimizing multiplication based on bits and decreasing the number of multiplies
- Seeing new multiplication algorithms.
- The Integer Multiplication Problem and its associated algorithms was most interesting.
- the different multiplication algorithms were hard to follow but still interesting
- I found Karatsubas Algorithm to be interesting, using the sum of all 4 multiplications to reduce total multiplications to 3 was cool
- How to find the ST-cuts and how max flow = min cut
- backedges as a way to undo previous selections
- Graph
- The maximum spanning tree used to do the PFS
- Ford Fulkerson applications

Assignment 4 Hint

- tripsWithin(budget)
  - Returns a set of paths whose total cost <= a given budget
  - Backtracking
  - Recursive helper method
  - Solve(current decision, current solution)
  - Current decision
    - Current vertex
  - Current solution
    - The set of paths so far
    - The current path
    - The remaining balance

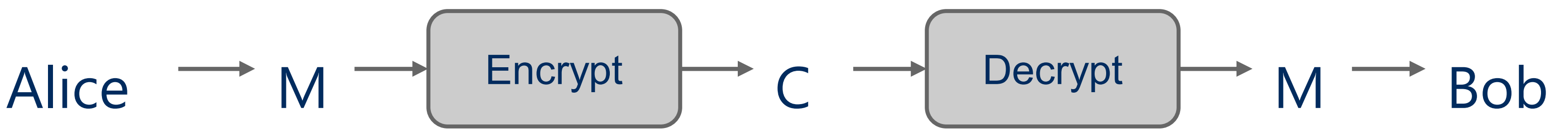
Assignment 4 Hint

Inside the recursive method

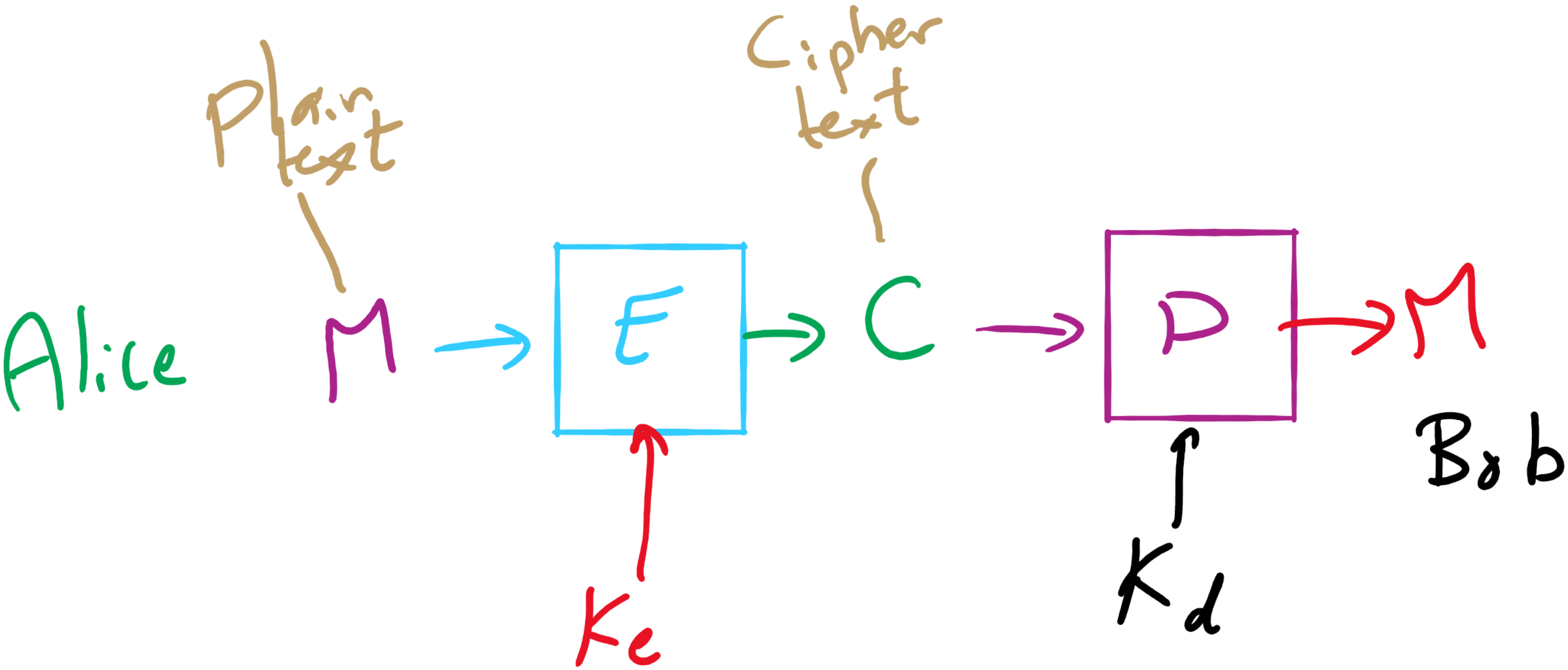
- Iterative over neighbors of current vertex
  - If neighbor is not marked and edge price <= remaining balance
    - Add neighbor to current path
    - Add **a copy of** current path to the set of paths
    - Mark the neighbor
    - Recur on the neighbor
    - Unmark the neighbor
    - Remove the neighbor from current path

Problem of the Day: Cryptography

- Cryptography - enabling secure communication in the presence of third parties
    - Alice wants to send Bob a message without anyone else being able to read it



Encryption Model



$K_e == K_d$

Symmetric

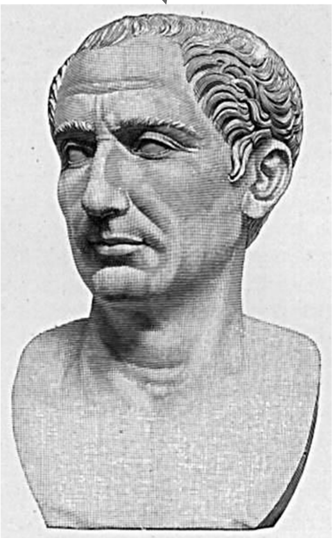
Enter the adversary

- Consider the adversary to be anyone that could try to eavesdrop on Alice and Bob communicating
  - People in the same coffee shop as Alice or Bob as they talk over WiFi
  - Admins operating the network between Alice and Bob
    - And mirroring their traffic to the NSA...
- Will have access to:
  - The *ciphertext*
    - The encrypted message
  - The encryption algorithm
    - At least Alice and Bob should assume the adversary does
- The key material is the only thing Bob knows that the adversary does not

Cryptography has been around for some time

- Early, classic encryption scheme:
  - Caesar cipher:
    - "Shift" the alphabet by a set amount
    - Use this shifted alphabet to send messages
    - The "key" is the amount the alphabet is shifted

Yes, that Caesar



Alphabet

ABCDEFGHIJKLMNOPQRSTUVWXYZ

XYZABCDEFGHIJKLMNOPQRSTUVW

Shift 3

By modern standards, incredibly easy to crack

- BRUTE FORCE
  - Try every possible shift
    - 25 options for the English alphabet
    - 255 for ASCII
- OK, let's make it harder to brute force
  - Instead of using a shifted alphabet, let's use a random permutation of the alphabet
    - Key is now this permutation, not just a shift value
  - R size alphabet means R! possible permutations!

By modern standards, incredibly easy to crack

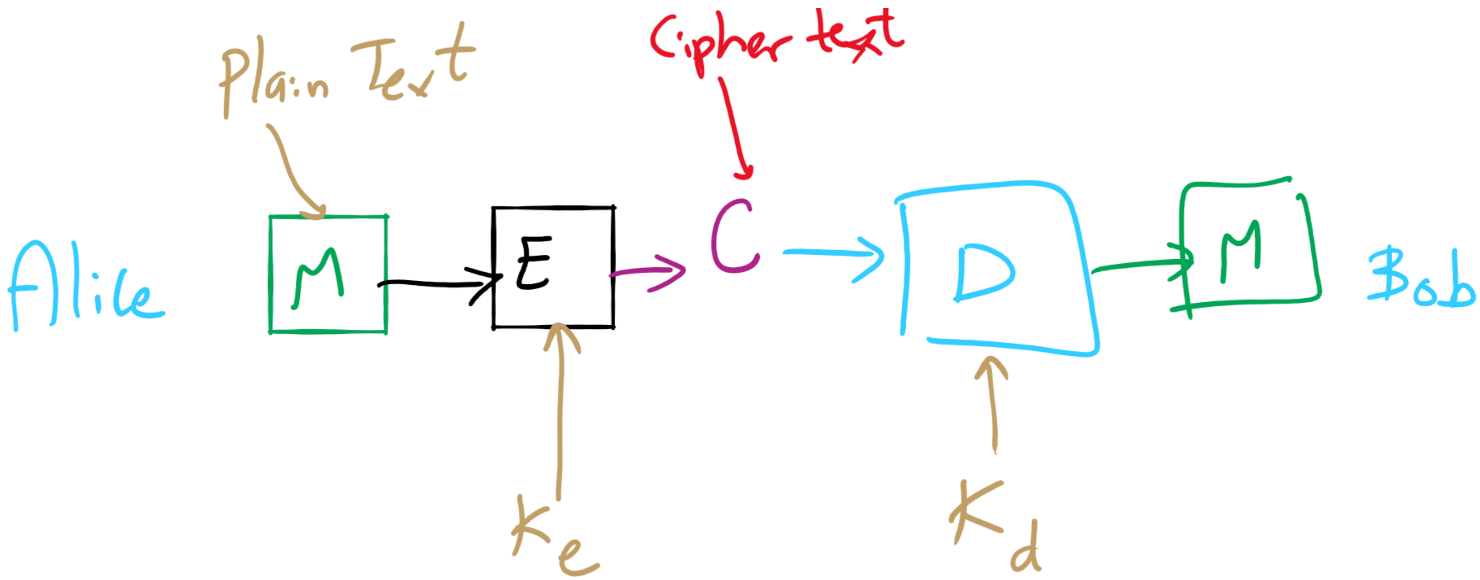
- Just requires a bit more sophisticated of an algorithm
- Analyzing encrypted English for example
  - Sentences have a given structure
  - Character frequencies are skewed
  - Essentially playing Wheel of Fortune

So what is a good cipher?

- One-time pads
  - List of one-time use keys (called a *pad*) here
- To send a message:
  - Take an unused pad
  - Use modular addition to combine key with message
    - For binary data, XOR
  - Send to recipient
- Upon receiving a message:
  - Take the next pad
  - Use modular subtraction to combine key with message
    - For binary data, XOR
  - Read result
- Proven to provide perfect secrecy

$$H(m \mid c) = H(m)$$

One-Time Pad



<u>One-time Pad</u>

random #'s

$K_e = K_d$ (single-use only)

E : modular addition (XOR)

D : modular subtraction (XOR)

One-time pad example

Encoding:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Message:

```
     H   E   L   L   O
     7   4  11  11  14
+   16   9   2  22  19    (mod 26)
------------------------------------
    23  13  13   7   7
```

Pad:

```
 Q   J   C   W   T
16   9   2  22  19
```

Encrypted Message:

```
     X   N   N   H   H
    23  13  13   7   7
-   16   9   2  22  19    (mod 26)
------------------------------------
     7   4  11  11  14
     H   E   L   L   O
```

One-Time Pad Example 1

Alice :

$$
\begin{array}{r}
101110011 \\
\oplus\ 111000101 \\
\hline
C = 010110110
\end{array}
$$

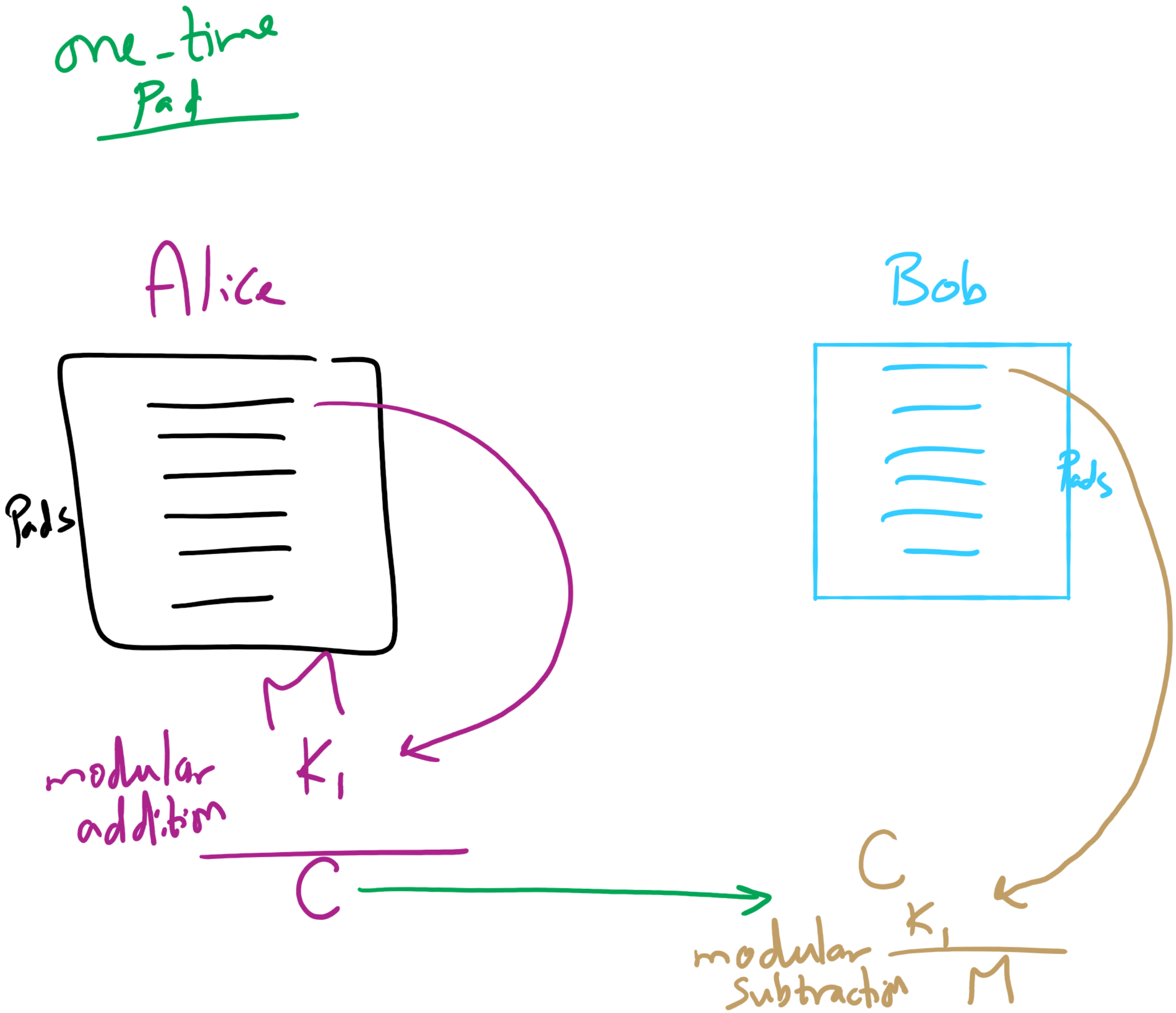
Bob :

$$
\begin{array}{r}
010110110 \\
\oplus\ 111000101 \\
\hline
101110011 = M \checkmark
\end{array}
$$

One-Time Pad Example 2

$$
\begin{array}{rl}
M: & 1101010 \\
K_1: \oplus & 1000111 \\
\hline
C: & 0101101
\end{array}
\quad \longrightarrow \quad
\begin{array}{rl}
 & 0101101 \\
\oplus & 1000111 \\
\hline
M: & 1101010
\end{array}
$$

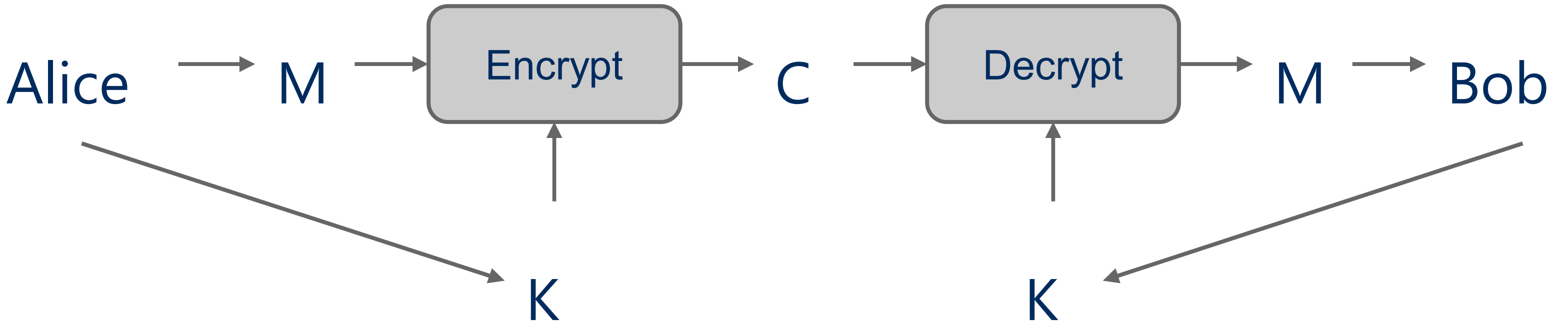
One-Time Pad


one-time
Pad
Alice
Pads
M
modular
addition
K_1
C
Bob
Pads
C
K_1
modular
subtraction
M

Difficulties with one-time pads

- Pads must be truly random
- Both sender and receiver must have a matched list of pads in the appropriate order
- Once you run out of pads, no more messages can be sent

Symmetric ciphers



- E.g., DES, AES, Blowfish
- Users share a single *key*
  - Numbers of a given bitlength (e.g., 128, 256)
  - Key is used to encrypt/decrypt many messages back and forth
- Encryptions/decryptions will be fast
  - Typically linear in the size the input
- Ciphertext should appear random
- Best way to recover plaintext should be a brute force attack on the encryption key
  - Which we have shown to be infeasible for 128bit AES keys

Problems with symmetric ciphers

- Alice and Bob have to both know the same key
  - How can you securely transmit the key from Alice to Bob?
- Further, if Alice also wants to communicate with Charlie, her and Charlie will need to know the same key, a different key from the key Alice shares with Bob
  - Alice and Danielle will also have to share a different key…
  - etc.

Enter public-key encryption

- Each user has their own pair of keys
  - A *public* key that can be revealed to anyone
  - A *private* key that only they should know
- How does this solve our problem?
  - Public key can simply be published/advertised
    - Posted repositories of public keys
    - Added to an email signature
  - Each user is responsible only for their own keypair
- Let's look at a public-key crypto scheme in detail...

RSA Cryptosystem in-depth

- What are RSA keypairs?
- How messages encrypted?
- How are messages decrypted?
- How are keys generated?
- Why is it secure?

RSA keypairs

- *Public* key is two numbers, which we will call n and e
- *Private* key is a single number we will call d
- The length of n in bits is the key length
  - I.e., 2048 bit RSA keys will have a 2048 bit n value
    - Note that "n" will be used to indicate the RSA public key component for our discussion of RSA...

Encryption

Say Alice wants to send a message to Bob

1. Looks up Bob's public key
2. Convert the message into an integer: m
3. Compute the ciphertext c as:
   - $c = m^e \pmod{n}$
4. Send c to Bob

Decryption

Bob can simply:

1. Compute m as:
    a. $m = c^d \pmod{n}$
2. Convert m into Alice’s message

Really?

What??!!

n, e, and d need to be carefully generated

1. Choose two prime numbers p and q
2. Compute $n = p * q$
3. Compute $\varphi(n)$
   - $\varphi(n) = \varphi(p) * \varphi(q) = (p - 1) * (q - 1)$
4. Choose e such that
   - $1 < e < \varphi(n)$
   - $GCD(e, \varphi(n)) = 1$
     - I.e., e and $\varphi(n)$ are co-prime
5. Determine d as $d = e^{-1} \bmod(\varphi(n))$

What the φ?

- Here, we mean φ to be Euler's totient
- φ(n) is a count of the integers < n that are co-prime to n
  - I.e., how many k are there such that:
    - 1 <= k <= n AND GCD(n, k) = 1
- p and q are prime..
  - Hence, φ(p) = p - 1 and φ(q) = q -1
- Further, φ is multiplicative
  - Since p and q are prime, they are co-prime, so
    - φ(p) * φ(q) = φ(p * q) = φ(n)
      - I won't detail the proof here...

$\phi(9) = ? = 6$

1, 2, 3, 4, 5, 6, 7, 8

$GCD(2, 9) = 1$

Greatest Common Divisor

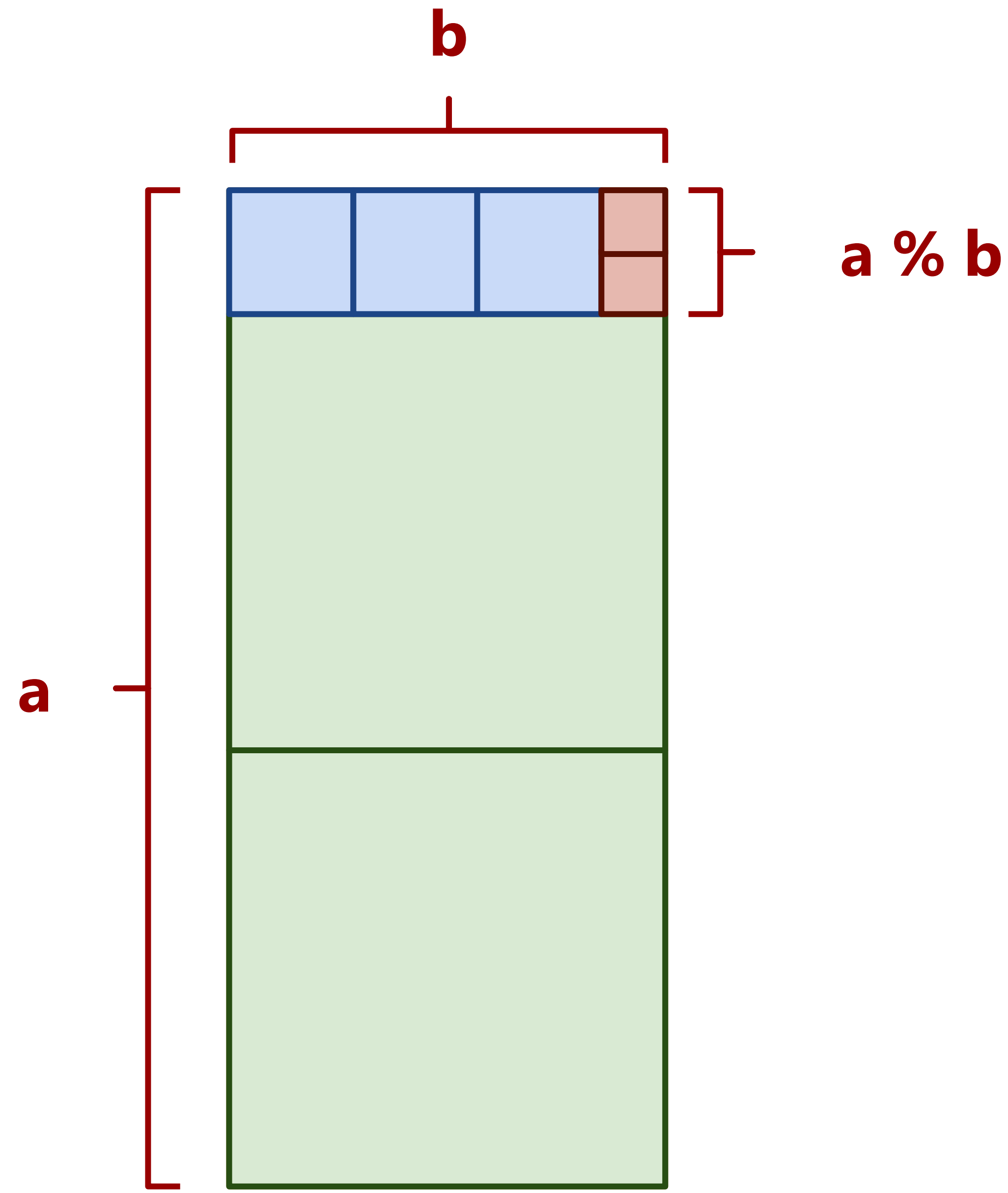
- GCD(a, b)
  - Largest int that evenly divides both a and b
- Easiest approach:
  - BRUTE FORCE

```
i = min(a, b)
while(a % i != 0 || b % i != 0):
        i--
```

- Runtime?
  - Θ(min(a, b))
  - Linear!
    - In *value* of min(a, b)...
  - Exponential in n
    - Assuming a, b are n-bit integers

Euclid's algorithm

- GCD(a, b) = GCD(b, a % b)



Euclidean example 1

- GCD(30, 24)
  - = GCD(24, 30 % 24)
- = GCD(24, 6)
  - = GCD(6, 24 % 6)
- = GCD(6, 0)…
  - Base case! Overall GCD is 6

Euclidean example 2

- = GCD(99, 78)
  - 99 = 78 * 1 + 21
- = GCD(78, 21)
  - 78 = 21 * 3 + 15
- = GCD(21, 15)
  - 21 = 15 * 1 + 6
- = GCD (15, 6)
  - 15 = 6 * 2 + 3
- = GCD(6, 3)
  - 6 = 3 * 2 + 0
- = 3

Analysis of Euclid’s algorithm

- Runtime?
  - Tricky to analyze, has been shown to be linear in n
    - Where, again, n is the number of bits in the input

Extended Euclidean algorithm

- In addition to the GCD, the Extended Euclidean algorithm (XGCD) produces values x and y such that:
  - GCD(a, b) = i = ax + by
- Examples:
  - GCD(30,24) = 6 = 30 * 1 + 24 * -1
  - GCD(99,78) = 3 = 99 * -11 + 78 * 14
- Can be done in the same linear runtime!

Extended Euclidean example

- = GCD(99, 78)
  - 99 = 78 * 1 + 21
- = GCD(78, 21)
  - 78 = 21 * 3 + 15
- = GCD(21, 15)
  - 21 = 15 * 1 + 6
- = GCD (15, 6)
  - 15 = 6 * 2 + 3
- = GCD(6, 3)
  - 6 = 3 * 2 + 0
- = 3

- 3 = 15 - (2 * 6)
- 6 = 21 - 15
  - 3 = 15 - (2 * (21 - 15))
  - = 15 - (2 * 21) + (2 * 15)
  - = (3 * 15) - (2 * 21)
- 15 = 78 - (3 * 21)
  - 3 = (3 * (78 - (3 * 21)))
    - (2 * 21)
  - = (3 * 78) - (11 * 21)
- 21 = 99 - 78
  - 3 = (3 * 78) - (11 * (99 - 78))
  - = (14 * 78) - (11 * 99)
  - = 99 * -11 + 78 * 14

GCD/XGCD Example 1

a = 63
b = 24

GCD(a,b) = GCD(b, a%b)

⟹ $2*24 + 15 = 63$ $\quad 15 = 63 - 2*24$
= GCD(24, 15)

GCD(24,15) = GCD(15, 24%15)

⟹ $1*15 + 9 = 24$ $\quad 9 = 24 - 1*15$
= GCD(15, 9)

$3 = x \cdot 63 + y \cdot 24$

GCD(15,9) = GCD(9, 15%9)

⟹ $1*9 + 6 = 15$ $\quad 6 = 15 - 1*9$
= GCD(9, 6)

GCD(9,6) = GCD(6, 9%6)

⟹ $1*6 + 3 = 9$
= GCD(6,3)

$2*3 + 0 = 6$
= GCD(3, 6%3)
= GCD(3,0)
= $\boxed{3}$

$$
\begin{aligned}
3 &= 9 - 1*6 \\
&= 9 - 1*(15 - 1*9) \\
&= 2*9 - 15 \\
&= 2*(24 - 1*15) - 15 \\
&= 2*24 - 3*15 \\
&= 2*24 - 3*(63 - 2*24)
\end{aligned}
$$

$$\boxed{3 = 8 * 24 - 3 * 63}$$

y $\qquad$ x

$$
\begin{array}{r}
192 \\
-\,189 \\
\hline
3
\end{array}
$$

GCD/XGCD Example 2

$GCD(\underset{a}{84}, \underset{b}{32})$

$GCD(a, b) = GCD(b, a \% b)$

$GCD(a, 0) = a$

$GCD(84, 32) = GCD(32, 84 \% 32)$

④ $84 = 2 * 32 + \underline{20}$ $\quad 20 = 84 - 2*32$

$= GCD(32, 20)$

$= GCD(20, 32 \% 20)$

③ $32 = 1 * 20 + 12$ $\quad 12 = 32 - 1*20$

$= GCD(20, 12)$

$= GCD(12, 20 \% 12)$

② $20 = 1 * 12 + 8$ $\quad 8 = 20 - 1*12$

$= GCD(12, 8)$

$= GCD(8, 12 \% 8)$

① $12 = 1 * 8 + 4$

$= GCD(8, 4)$

$= GCD(4, 8 \% 4)$

$= GCD(4, 0)$

$= \boxed{4}$

$4 = 12 - 1 * 8$

$= 12 - 1 * (20 - 1 * 12)$

$= 2 * 12 - 1 * 20$

$= 2 * (32 - 1 * 20) - 1 * 20$

$= 2 * 32 - 3 * 20$

$= 2 * 32 - 3 * (84 - 2*32)$

$= 8 * 32 - 3 * 84$ (y = 8, x = −3)

$4 = x * 84 + y * 32$

$$\begin{array}{r} 256 \\ -252 \\ \hline 4 \end{array}$$

OK, now what about multiplicative inverses mod φ(n)?

$1 = \phi(n)(-z) + e \cdot d$

$GCD(\phi(n), e) = \phi(n)(-z) + e \cdot d$

$x \qquad y$

- $d = e^{-1} \bmod(\varphi(n))$
- Means that $d = 1/e \bmod(\varphi(n))$
- Means that $e * d = 1 \pmod{\varphi(n)}$
- Now, *this* can be equivalently stated as $e * d = z * \varphi(n) + 1$
    - For some z
- Can further restate this as: $e * d - z * \varphi(n) = 1$
- Or similarly: $1 = \varphi(n) * (-z) + e * d$
- How can we solve this?
    - Hint: recall that we know $GCD(\varphi(n), e) = 1$

Use extended Euclidean algorithm!

- GCD(a, b) = i = ax + by
- Let:
  - a = φ(n)
  - b = e
  - x = -z
  - y = d
  - i = 1
- GCD(φ(n), e) = 1 = φ(n) * (-z) + e * d
- We can compute d in linear time!

RSA keypair example notes

- p and q must be prime
- $n = p * q$
- $\varphi(n) = (p - 1) * (q - 1)$
- Choose e such that
  - $1 < e < \varphi(n)$ and $GCD(e, \varphi(n)) = 1$
- Solve $XGCD(\varphi(n), e) = 1 = \varphi(n) * (-z) + e * d$
- Compute the ciphertext c as:
  - $c = m^e \pmod{n}$
- Recover m as:
  - $m = c^d \pmod{n}$

Please submit your reflections by using the CourseMIRROR App

If you are having a problem with CourseMIRROR, please send an email to coursemirror.development@gmail.com

8/29/2022